



## ZUM THEMA

### Komponentenbasierte Projektentwicklung

Wer heutzutage noch mithilfe einer Programmiersprache ein Programm verfasst, führt im Grunde eine Tätigkeit aus, die sich mittlerweile überlebt hat und zu einer geschichtlichen Epoche des Einsatzes von Computern gehört. Menschen, die am PC sitzen, brauchen heute von Programmierung keine Ahnung mehr zu haben, denn sie rufen nur noch ein Programm auf, mit dem sie ihr Problem – hoffentlich adäquat – lösen können. Zwar ist dieses Programm auch einmal codiert, d. h. im engeren Sinne programmiert worden, aber selbst dies ist nicht unbedingt vorauszusetzen. Denn für eine nahezu unübersehbare Fülle an Funktionen existieren Software-Bausteine – so genannte Komponenten –, die „nur“ noch von den Entwicklern mit einem *glue code*, einem verbindenden Programmcode „zusammengeklebt“ werden müssen. Was dies für den Informatikunterricht bedeutet, soll in diesem Heft erörtert werden.

Das Titelbild zum Thema wurde von Jens-Helge Dahmen, Berlin, für LOG IN gestaltet.

Impressum	2
Editorial	3
Berichte	4

## THEMA

Komponentenbasierte Softwareentwicklung von Jürgen Müller	18
Hallo Webserver! – Ein Einstieg in die dynamische Entwicklung von HTML-Seiten von Alfred Hermes	26
COM – Praktische Annäherung an ein Komponentenmodell von Jürgen Müller	34

## DISKUSSION

Objekte, Klassen, Module, Kontrakte und Komponenten von Harro von Lavergne	46
--	----

## PRAXIS & METHODIK

Objektorientierung im Anfangsunterricht!!! – Simulation einer Taschenlampe mit DELPHI von Christian Steinbrucker	56
Suchbaum-Modellierung – Mit Unterrichtsbeispielen in JAVA von Gerhard Röhner	62
Rekonstruktives Modellieren – Vorschläge für den Computereinsatz von dynamischen Geometriesystemen von Heinz Schumann	70
Zelluläre Drei-Farben-Automaten von Alfred Hermes	78
Virtuelle Ameisenwelt (Teil 2) von Paola Prätorius	81

Werkstatt: Interaktion und Kommunikation zwischen Objekten von Alfred Hermes	90
---	----

## SCHULEN ANS NETZ

<i>eTwinning</i> bei Schulen ans Netz von Bernhard Koerber	97
--	----

## COLLEG

Unterricht mit StarOffice 7 (Teil 2) von Ingo-Rüdiger Peters und Bernhard Koerber	99
---	----

## COMPUTER & ANWENDUNGEN

Multimedia	109
Software: UMLed – Ein Software-Werkzeug für den Unterrichtseinsatz	110
Generische Typen – Neu in JAVA 5.0	112
Geschichte	114
Online	120

## FORUM

Rezension: Reichert, R.; Nievergelt, J.; Hartmann, W.: Programmieren mit Kara	121
Hinweise auf Bücher	123
Medien	124
Info-Markt	126
Veranstaltungskalender	126
Computer-Knobelei: Kringel, Kreuze und Klein	127
Vorschau	128
LOG OUT	128

81 <b>Beilage:</b> Jahresregister LOG IN 2004	
---	--



# Bausteine als Software

Aus den Holzbausteinen eines Bausteinkastens schufen Kinder früherer Zeiten Häuser, Ställe, Burgen, ganze Dörfer. Und selbst im Zeitalter der LEGO-Steine entstehen in den Kinderstuben immer noch fantasievolle Gebäude. „Ein Baustein“, so definiert es *Wikipedia*, die freie Enzyklopädie, „ist ein Teil eines größeren Ganzen, der für sich alleine genommen möglicherweise keine oder nur eine geringe, als Teil dieses Ganzen jedoch eine beträchtliche Wichtigkeit hat.“

Mittlerweile existiert der Begriff *Baustein* auch im übertragenen Sinn, und es wird damit alles bezeichnet, was zusammen mit anderen Bausteinen zu einem größeren Objekt wird. Dass deshalb beim ingenieurmäßigen Bau von Software, im *Software Engineering*, auch von Bausteinen gesprochen wurde, liegt auf der Hand. Und deshalb hat sich LOG IN bereits im Heft 6/1989 mit dem Thema „Software-Bausteine“ auseinandergesetzt.

Aber das Wort *Baustein* erinnert wohl doch zu sehr an Holzklötzchen. Deshalb wird in der englisch sprechenden Welt natürlich auch nicht *building brick* zu einem Software-Baustein gesagt, sondern *component*. Und so ist der Begriff Software-Baustein aus der Mode gekommen und durch *Komponente* ersetzt worden. Eigentlich stammt der Begriff *Komponente* aus dem Lateinischen – *componere* bedeutet auf Deutsch so viel wie *zusammenstellen* oder *zusammensetzen*.

In der Softwaretechnik wird der Begriff *Komponente* allerdings unterschiedlich definiert. Manche verstehen darunter lediglich Quellcode, der in eine umfangreichere Software integriert wird, bzw. Softwarebibliotheken, die in Entwicklungsprojekten verwendet werden. Unterprogramme bzw. Prozeduren könnten ebenfalls als Komponenten verstanden werden. Heute bezeichnen Softwaretechniker als *Komponente* einen funktional oder konstruktiv zusammengehörigen, abgeschlossenen Bestandteil eines Systems.

Die Tätigkeit des Programmierens hat sich durch die Idee, ein Programm komponentenbasiert zu erstellen, grundlegend gewandelt: Wo vorher ein einziges Programm zentral entwickelt wurde – ggf. mit einigen Prozeduren –, werden heute Programme aus präzise definierten Komponenten zusammengesetzt, und das Hauptprogramm besteht oft „nur“ noch aus einem *Graphical User Interface*, einer grafischen Benutzungsoberfläche. Doch es geht noch weiter: Zunehmend rückt das Modellieren von Anwendungen in den Mittelpunkt der Softwareentwicklung und verdrängt das „klassische“ Programmieren, das man gerne an die Maschine übergeben möchte. Bei diesem Prozess soll ein einfach zu verstehendes Fachmodell in mehreren Schritten in ablauffähigen Code transformiert werden. Dabei spielt die automatisierte Generierung von Softwarekomponenten eine zentrale Rolle. Dieser Gedanke einer modellgetriebenen Entwicklung erfreut sich unter dem Begriff *Model Driven Architecture* (MDA) auf Fachkongressen einer hohen Beliebtheit. Eine erste Unterstützung durch Software gibt es bereits; Hersteller von Entwicklungsumgebungen wie *Borland* und *Microsoft* vermarkten beispielsweise Werkzeuge zur Generierung von Code aus UML-Modellen.

Gehört es dann überhaupt noch zu den Bildungsaufgaben der Schule, Schülerinnen und Schülern im Informatikunterricht das Programmieren beizubringen? Die Nutzung von komponentenbasierten Entwicklungsumgebungen wie DELPHI, ECLIPSE oder VISUAL STUDIO im Unterricht ist sicherlich zwiespältig. Denn einerseits gibt es Vorteile:

- ▷ Eine moderne Programmierumgebung kann motivierend auf die Schülerinnen und Schüler wirken.
- ▷ Die Einführungsphase dauert zwar länger, aber später geht's im Unterricht schneller voran, und die Ergebnisse sehen „professioneller“ aus.

- ▷ Leistungsstarke Schülerinnen und Schüler können durch die vielfältigen Funktionalitäten der komponentenorientierten Entwicklungsumgebungen besser mit Zusatzaufgaben versorgt werden, sodass das differenzierte Arbeiten und das Arbeiten in Arbeitsgruppen besser unterstützt werden kann.
- ▷ Die Welt der Objektorientierung wird den Schülerinnen und Schülern intuitiv vermittelt.

Andererseits dürfen aber die Probleme nicht übersehen werden:

- ▷ Es besteht eine hohe Ablenkungsgefahr durch die vielen Funktionalitäten, die Entwicklungsumgebungen bieten.
- ▷ Die Programmierung mit solchen Umgebungen verführt zur Spielerei mit der Oberfläche – die algorithmische Idee könnte in den Hintergrund treten.
- ▷ Die Bedienung von Umgebungen wie DELPHI u.a. ist für Anfänger relativ kompliziert, sodass sie sich schnell überfordert fühlen.
- ▷ Eine Menge Details und technische Einzelheiten müssen entweder vom Lehrer erläutert werden – oder Schülerinnen und Schüler akzeptieren diese unverstanden und arbeiten mechanisch weiter.
- ▷ Als Ergebnis der Arbeit entsteht nicht nur eine einzige Programmdatei, sondern es entstehen viele verschiedene Dateien, die Komponenten, Konfigurationsdaten u. v. a. m. enthalten.
- ▷ Das Beschaffen der Software ist in der Regel nicht billig (Open-Source-Produkte wie ECLIPSE natürlich ausgenommen).

Die Didaktik der Informatik betritt hier in gewisser Weise Neuland. Deshalb sollen in diesem Heft einige Anregungen gegeben werden, auch im Unterricht das komponentenbasierte Entwickeln von Software zu erproben.

Bernhard Koerber  
Jürgen Müller